LISTING OF CLAIMS

1. (Previously Presented) A computer-implemented method comprising:

automatically detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem;

determining whether the change is compatible with the software objects of the second software subsystem; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

2. (Previously Presented) The method of claim 1 wherein determining whether the change is compatible further comprises determining whether the change is predefined as compatible.

3. (Original) The method of claim 2 further comprising allowing the change if the change is predefined as compatible.

4. (Original) The method of claim 3 further comprising issuing a message that the change is not allowed if the change is not predefined as compatible.

5. (Original) The method of claim 4 further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

6. (Previously Presented) A computer-implemented method comprising:

identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen;

detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change, determining with a compatibility check whether the change is compatible with a second software subsystem; and

issuing a notice indicating results of the compatibility check.

7. (Original) The method of claim 6 wherein the subset of software objects declared frozen includes software objects of the first software subsystem that are used by the second software subsystem.

8. (Previously Presented) The method of claim 7 wherein frozen software objects are classified to include released objects and restricted objects.

9. (Previously Presented) The method of claim 8 wherein the released objects include software objects that are used by the second software subsystem without restrictions.

10. (Previously Presented) The method of claim 8 wherein the restricted objects include software objects that are used by software objects of the second software subsystem, the software objects being fewer in number than a threshold.

11. (Previously Presented) The method of claim 8 wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object.

12. (Previously Presented) The method of claim 8 wherein classification of the frozen software objects is based on a number of times a frozen software object is used by the second software subsystem.

13. (Original) The method of claim 6 wherein a software object is a function module.

14. (Original) The method of claim 6 wherein a software object is a data structure.

15. (Original) The method of claim 13 wherein the software object includes an environment of the function module.

16. (Original) The method of claim 6 wherein a software object includes a class and an environment of the class.

17. (Original) The method of claim 6 wherein a software object includes an interface and an environment of the interface.

18. (Original) The method of claim 6 wherein a software object includes a program and an environment of the program.

19. (Original) The method of claim 6 wherein the detecting the change comprises automatically monitoring development of software code.

20. (Original) The method of claim 6 wherein the determining whether the change is compatible comprises determining whether there is a predefined declaration of compatibility of the change.

21. (Original) The method of claim 7 wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

22. (Original) A computer-implemented method comprising:

performing a global compatibility check of software objects of a first software subsystem by determining whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were introduced without obtaining prior approval;

identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system; and issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change.

23. (Previously Presented) The method of claim 22 wherein the performing a global compatibility check comprises comparing a current version of software code with a version of the software code at the time of a last global compatibility check.

24. (Original) The method of claim 22 wherein the subset of the software objects includes frozen software objects.

25. (Original) The method of claim 24 wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem.

26-31. (Canceled)

32. (Previously Presented) An article of manufacture comprising:

a storage medium having stored therein instructions which, when executed by a processor, cause a processing system to perform a method comprising;

detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem;

determining whether the change is compatible with the software objects of the second software subsystem; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

33. (Previously Presented) The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method wherein determining whether the change is compatible further comprises determining whether the change is predefined as compatible.

34. (Previously Presented) The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method further comprising issuing a notification that the change is not allowed if the change is not predefined as compatible.

35. (Previously Presented) The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

36-38. (Canceled)